LESSON 4 – BASIC MACHINE LEARNING APPLICATIONS

Dr. Jack Hong
Adjunct Faculty, Lee Kong Chian School of Business, SMU
Co-founder, Research Room Pte. Ltd.

INTRODUCTION TO MACHINE LEARNING

INTRODUCTION

- We will not cover all machine learning methods, but only those that are most effective and widely used
  - There are too many algorithms out there!
- Machine learning algorithms are classified into 2 categories:
  - Supervised learning
  - Unsupervised learning

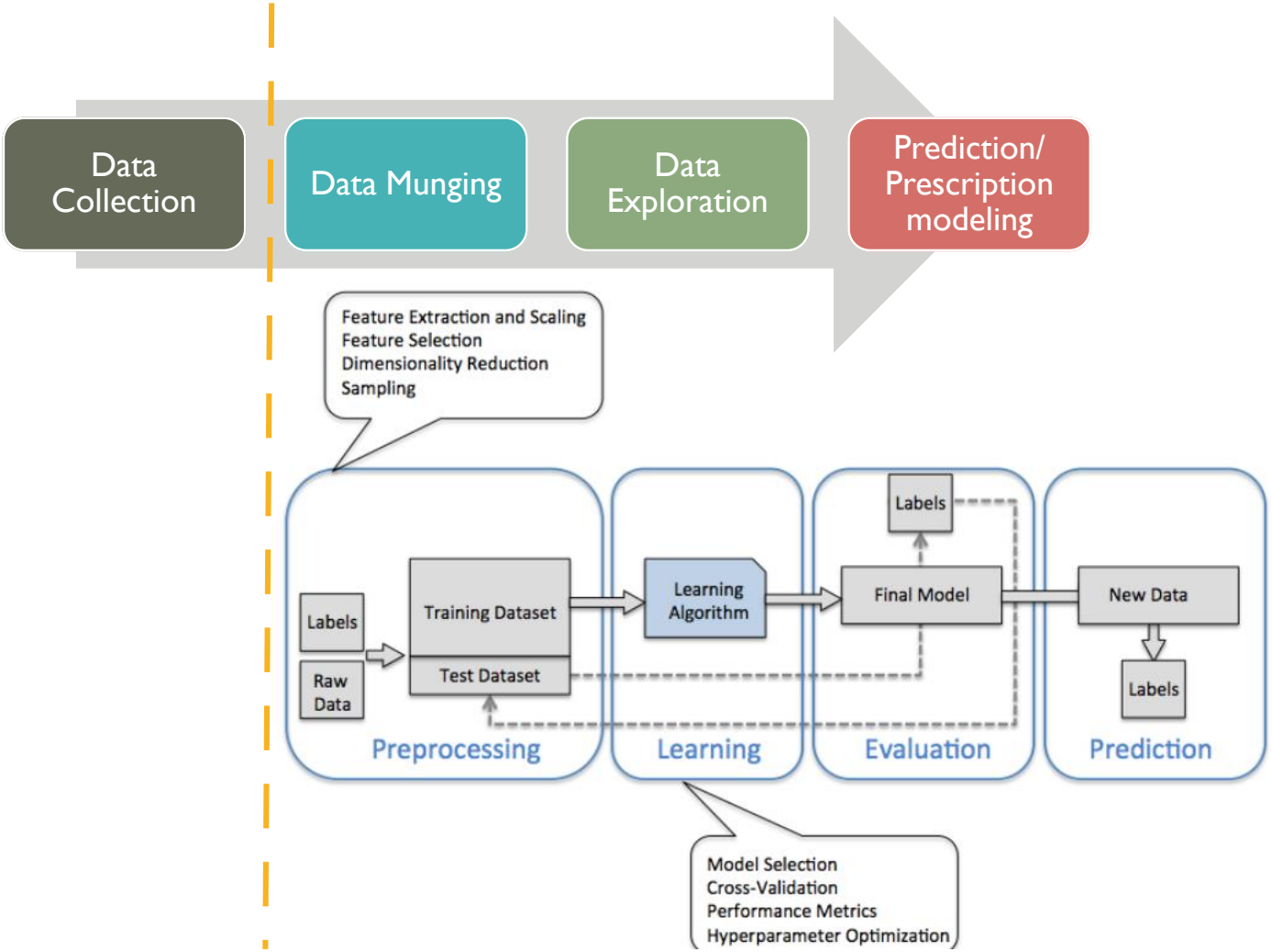
SUPERVISED LEARNING

- Train a model to learn from labeled training data
  - Each observation is classified with an outcome (e.g. Spam or Not Spam) or a continuous variable (as in regression)
  - We extract features from each observation
    - In the above example, the words that occur in each email
    - Can also include other variables such as time, name of sender, title etc
    - Compute a probability of outcome (spam or not spam) associated with the presence and absence of the features
    - In the case of the continuous variable, to minimize the loss function
- Reinforcement learning is about improving a model's performance based on its interaction with the environment
  - Uses reward function compared to loss function
  - E.g. Chess engine: System decides moves based on the state of the board, and the reward function will determine win or lose as the outcome

UNSUPERVISED LEARNING

- In supervised learning, we know the answer beforehand
- In unsupervised learning, we deal with unlabeled data
  - Unsupervised learning techniques allow us to explore the data structure to extract information without any prior guidance
  - Clustering: grouping observations by similarity and dissimilarity
  - Dimensionality reduction: removing noise from data while compressing high dimensional data into smaller ones

WHERE DOES MACHINE LEARNING FIT IN THE DATA SCIENCE VALUE CHAIN?



DATA MUNGING IN PYTHON

DATA MUNGING IN PYTHON

- Data munging is a necessary but dreaded process in data analytics
  - Involves cleaning, transforming, and setting data the right way
- Compared to R, Python is more powerful (and convenient) in data munging
  - Data scientists use the pandas package for this purpose
  - Pandas is developed by Wes McKinney, a Financial Economist
  - a financial economist to handle economics and finance datasets (refer to panel data setup in Lesson 3)
    - Pandas got its name not from the animal but because it was meant for handling **pan**el **da**ta.
  - The package was so useful that its now the de-facto data munging package in Python

IMPORTANT TECHNIQUES USING PANDAS AND SCIKIT-LEARN

- I'll cover a few key techniques that you will use frequently
  - Upload "ML introduction.ipynb" and "wine.csv" into your Jupyter dashboard
  - Follow the comments and instructions
  - This notebook tutorial covers data munging and predictive modeling

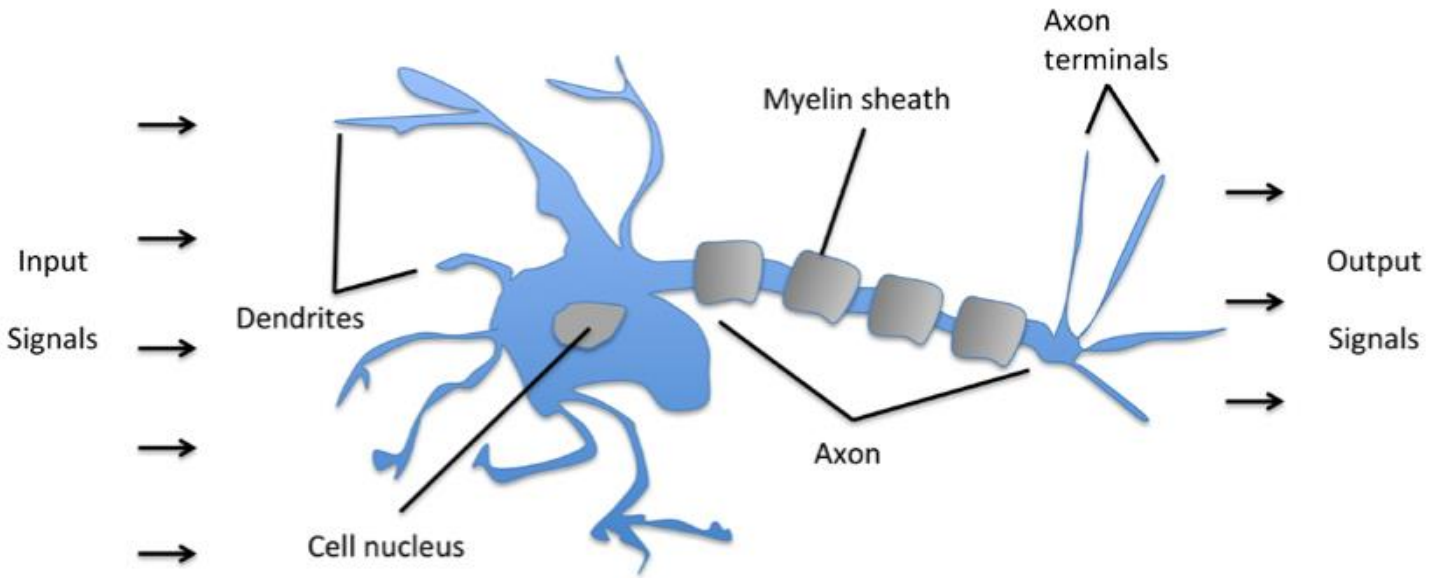
- To know more about pandas, you may refer to https://pandas.pydata.org/pandas-docs/stable/10min.html for the official 10 minutes guide to pandas

PERCEPTRON LEARNING

WHERE IT ALL STARTED FOR MACHINE LEARNING

THE BEGINNINGS OF MACHINE LEARNING

- Early works in machine learning attempts to mimic how a brain cell works, in simplistic forms
  - Simple logic gate with binary outputs, triggered by the level/strength of accumulated signal against a threshold

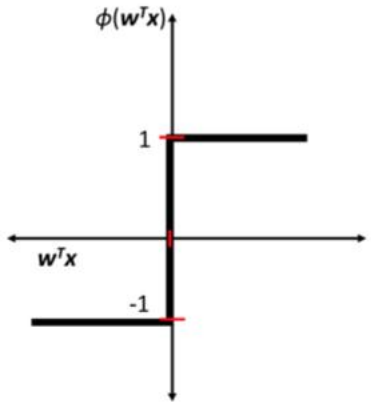


PERCEPTRON LEARNING – BASIC ML ILLUSTRATION

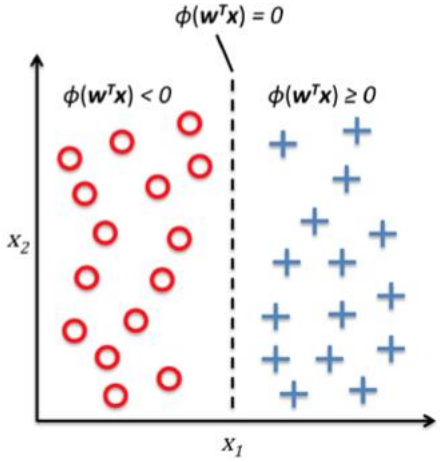
- Based on the Neuron model, Frank Rosenblatt conceptualized a learning rule that would
  - Automatically learn the optimal weight coefficients
  - Multiplied with the input features
  - Decide whether a neuron fires or not
- This is akin to a classification task
- In the below example, we use a step function (if x > 0.5, y = red) (else y = blue)
  - All values of x will then be mapped into a y-value of 1 or 0
  - Example: If humidity is > 60%, it will rain, else it will not

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

weights **input**

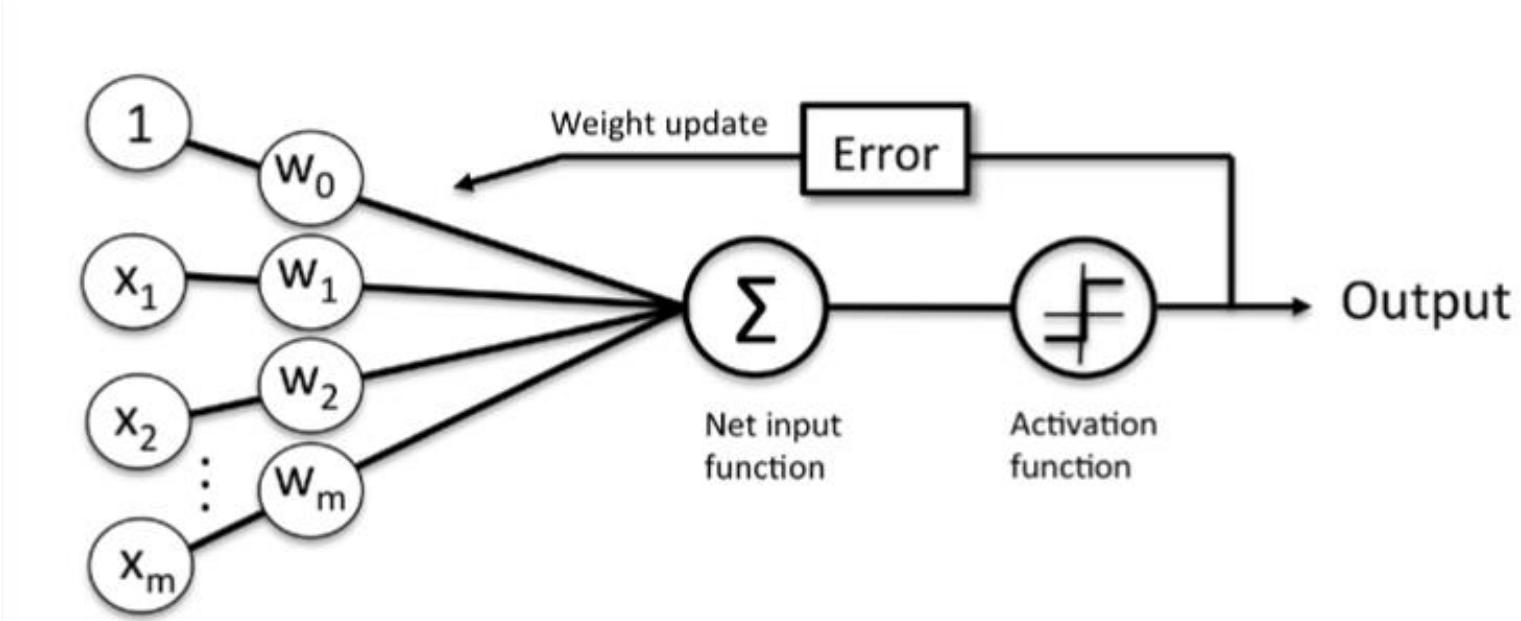


activation function



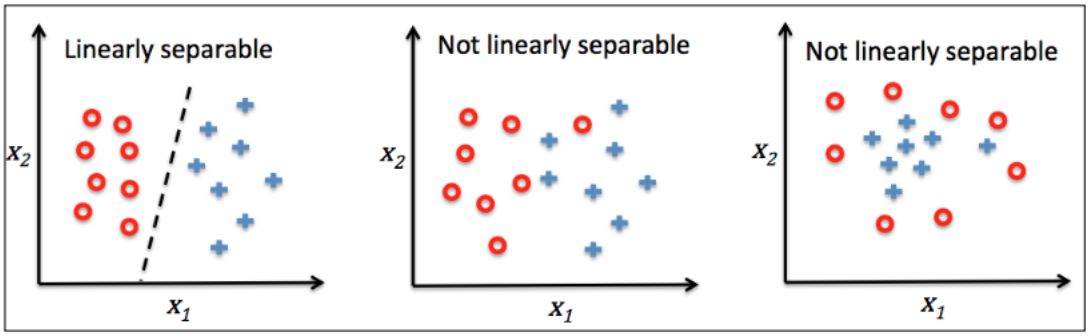
HOW DOES THE MODEL LEARN?

- An ML model learns by iteration
- It chooses a set of starting weights for each feature input
- Throws it through the activation function and compares the predicted output against the actual
- The model continuously attempts to reduce the error (actual – predicted) by changing the weights for the input features



PERCEPTRON LEARNING

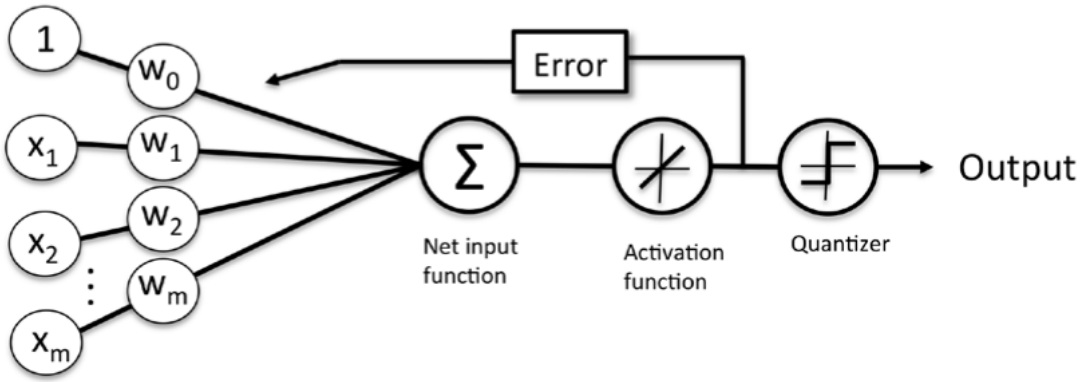
- Intuitively, we can see that this method adopts a linear separation mechanism to split samples via a boundary
  - However, if the samples cannot be perfectly separated, then the perceptron will never terminate
  - The solution is to allow some kind of error threshold (e.g. 20% misclassification)



- We can generalize this perceptron model and change the step function to other functional forms
  - Such as the linear and logistic models we learnt in Lesson 3
- Machine learning is very closely related to statistical modeling

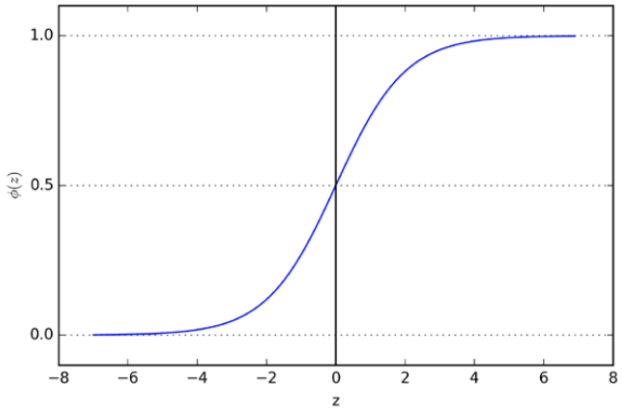
LINEAR REGRESSION IN MACHINE LEARNING

- The linear regression model is implemented in machine learning by using a linear activation function
- This ML model is known as Adaline

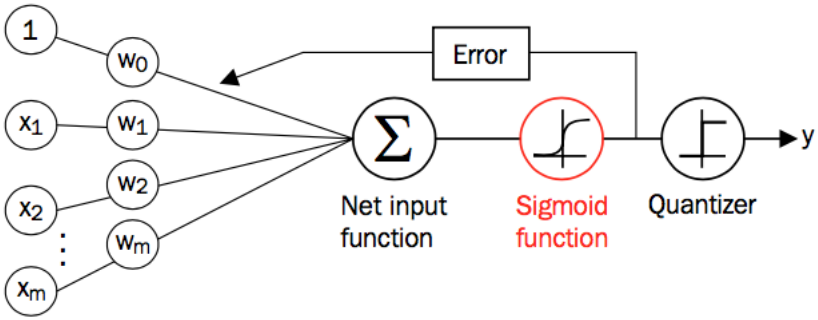


LOGISTIC REGRESSION IN MACHINE LEARNING

- Recall that logistic regression uses a logit function to map values of x to a range bounded by (0, 1)
- In machine learning, such a function is known as the sigmoid function due to its S-shape
- Implementing the logistic regression in ML is simply using the sigmoid function as the activation function



Sigmoid function



GRADIENT DESCENT I

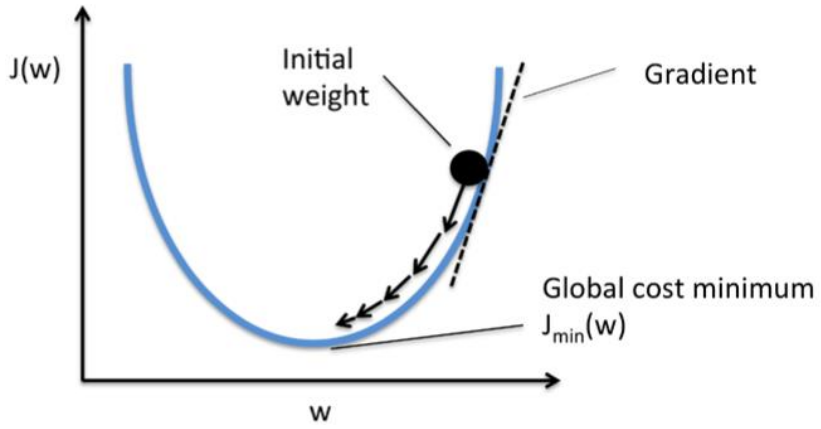
- By changing the activation function to a linear function, we are attempting to fit data into a linear form, similar to the linear regression
  - The model error is now computed based on a linear prediction
- Like regression models, the objective of the ML algorithm is to find a set of weights for each input feature that minimizes predictive errors (mathematically, its defined as the sum of squared errors)

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_i \left(y^{(i)} - \phi\left(z^{(i)}\right)\right)^2$$

- This is the general principle of supervised machine learning algorithms:
  - Define an objective function to be optimized by reducing the error between predicted and actual values
  - Shift the weights to find the minimum error rate achievable (in machine learning speak, we also call error the cost)

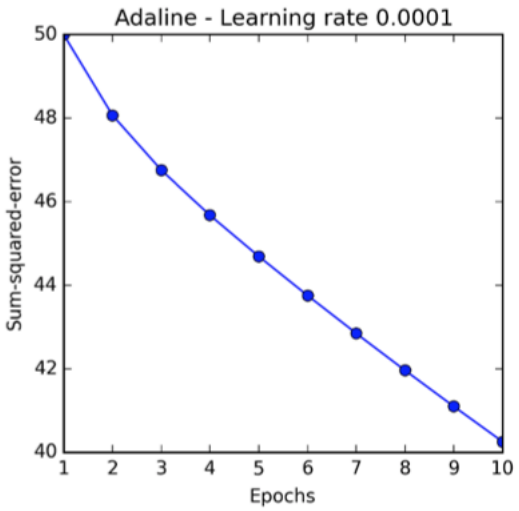
GRADIENT DESCENT II

- Analogous to climbing down a hill to reach the bottom of the valley

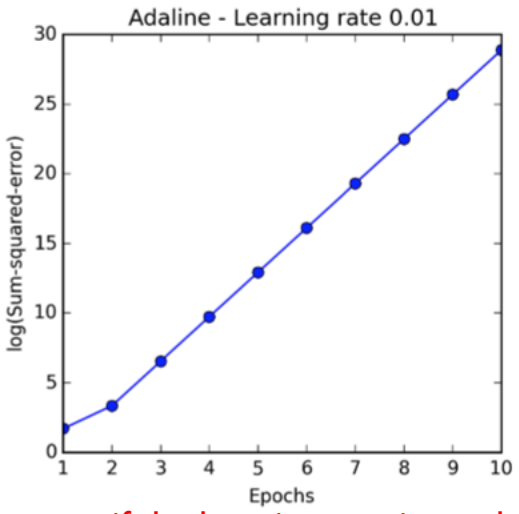


- Hyper-parameters
  - For the above model, there are 2 parameters that we need to care about as we perform the gradient descent
    - Number of iterations: How many times should we move the weights around?
    - Learning rate: How much movement should each iteration allow?

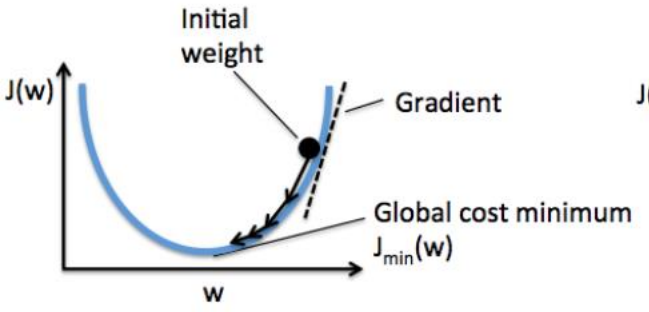
CHOOSING HYPERPARAMETERS

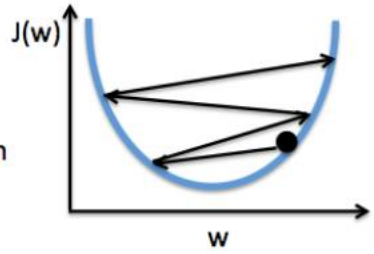


A small learning rate requires a large amount of iterations to converge



However, if the learning rate is too large, we keep overshooting the global minimum and errors increase



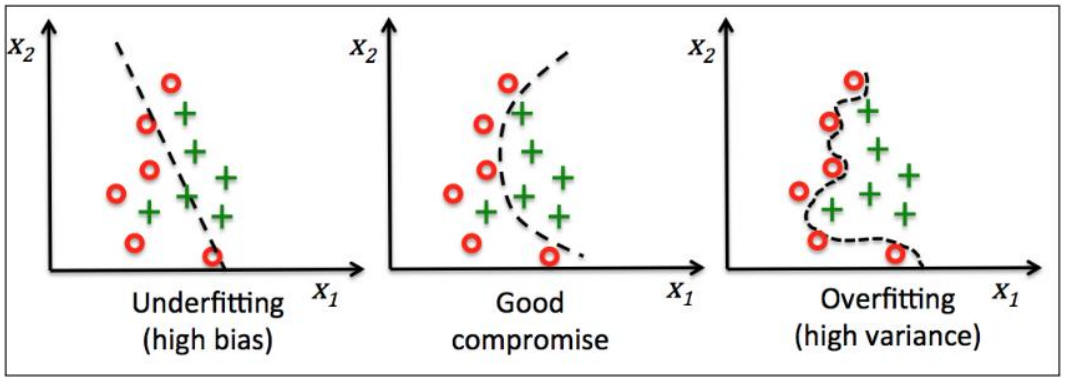


- Hyperparameter tuning (finding the best hyperparameter) is an indispensable step in machine learning modeling
  - There are many tricks to hyperparameter tuning but we will not cover them in this course

PREDICTIVE ACCURACY

OVERFITTING AND UNDERFITTING

- Recall in Lesson 3, we had to split the dataset into training and test set
  - Because a model tends to perform well on data it has "seen before" (in-sample data)
  - This has an upward bias on reported predictive accuracy
    - Performs well on training data, but badly on out-sample data
  - The solution is to train the model in-sample and test it on out-sample data
  - The name of this problem is known as "overfitting"
- In contrast to overfitting, underfitting is another issue where the model is too naïve (e.g. too little features) to capture significant patterns in the training data
  - Reported predictive accuracy is weak for both in and out-sample data



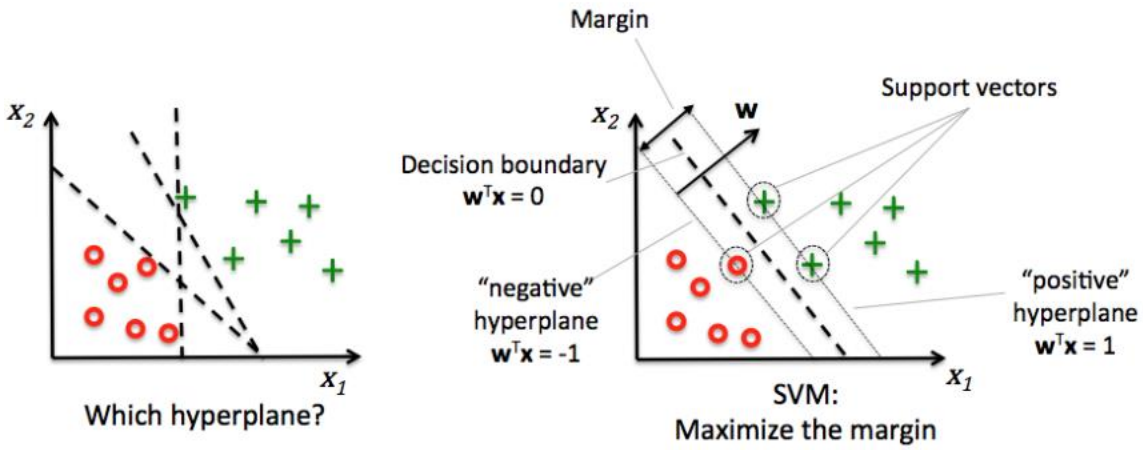
REGULARIZATION TO COMBAT OVER AND UNDER FITTING

- Finding a good trade-off requires some form of tuning on the complexity of the model
- Regularization is a good method for this purpose
  - Can handle collinearity and filter noise
  - Introduce bias to penalize extreme parameter weights
  - Called L1 or L2 regularization, or a mix of both
    - Most commonly used: L2 regularization (also called shrinkage or weight decay)
- Regularization parameters are included as hyperparameters in most ML algorithms in Python's scikit-learn package

SUPPORT VECTOR MACHINES

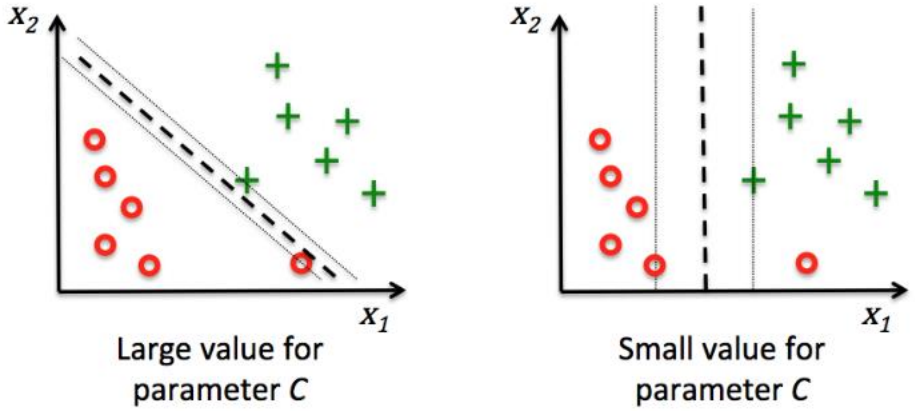
SUPPORT VECTOR MACHINES (SVM)

- A popular and powerful ML algorithm extended from the Perceptron model
- Recall the optimization objective of the Perceptron model is to minimize errors
  - SVM's optimization objective is to maximize the distance between observations separated by a decision boundary called the hyperplane
    - In other words, SVM optimizes by maximizing the margin
    - Note that hyperplanes can be multi-dimensional (which is often the case)



SEPARATION WITH ERRORS

- In the event that a perfect cut cannot be established, the SVM model allows for a threshold of misclassification errors
  - In the below example, the parameter C defines this threshold



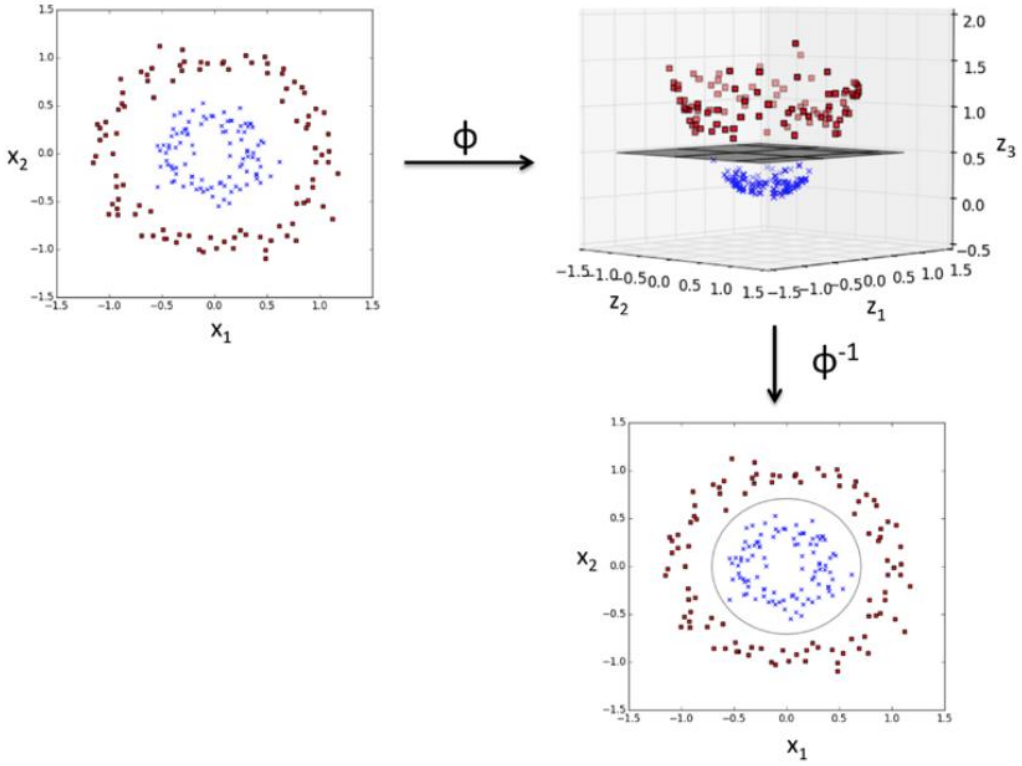
The value of C corresponds to the size of the penalties for misclassification errors

SVM FOR NONLINEAR PROBLEMS

- Kernel SVM
  - Used when its not possible for us to separate positive and negative outcomes using linear methods
  - The idea is to create nonlinear combinations of the original features/variables and project them onto a higher dimensional space via a mapping function
    - then we can do a linear hyperplane in this higher dimensional space
    - map it back to the original lower dimensional space after we are done

MAPPING FUNCTION

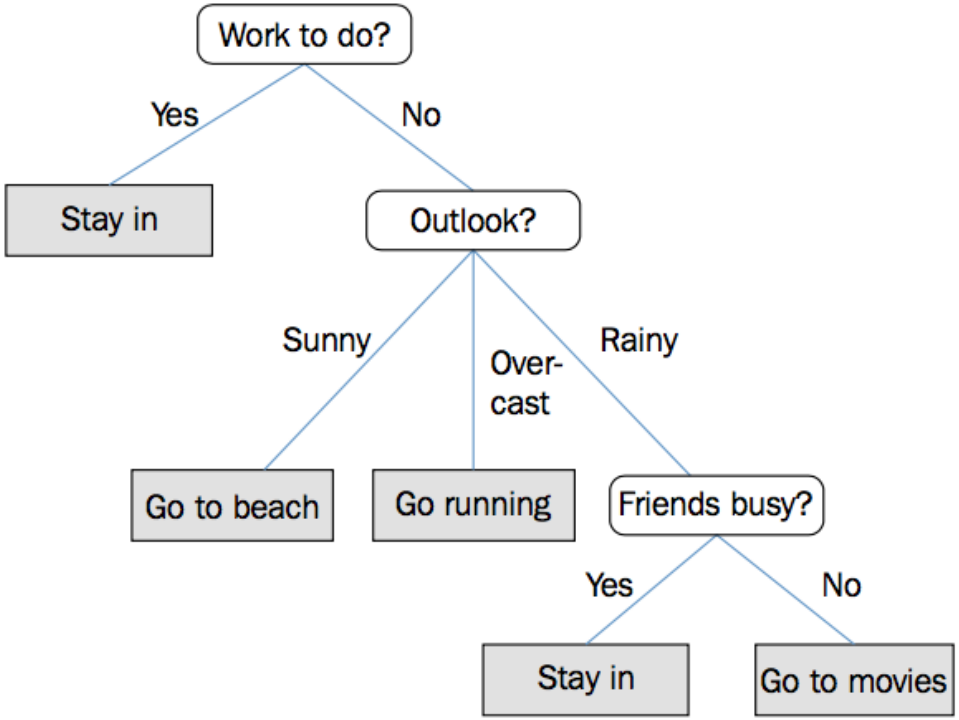
- Example:
  - Transform a 2-dimensional dataset into a 3-dimensional one
    - In this case, the closer the values of X_1 and X_2 are to the center, the lower the Z value is
  - Use a linear function to cut a line between the red and blue observations
  - Transform the results back into 2-d



DECISION TREES

DECISION TREES

- Decision trees are so named because the model breaks the data down by decision-based questions
  - Features can be categorical, as well as continuous



IMPLEMENTING DECISION TREE LEARNING

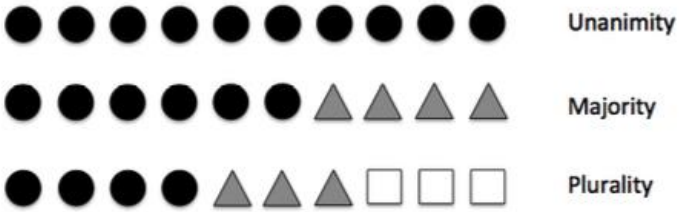
- The splitting rule at each node of the tree acts on the feature that results in the largest Information Gain (IG)
  - This is repeated at each child node until the observations at each node all belong to the same class
- As such, decision tree models are extremely good at fitting
  - The model can create, shrink, and expand boxes to envelop datapoints with the same outcome variable
  - However, such a procedure is prone to overfitting
    - Techniques such as pruning (sets a limit for the maximal depth of the tree) help to mitigate this

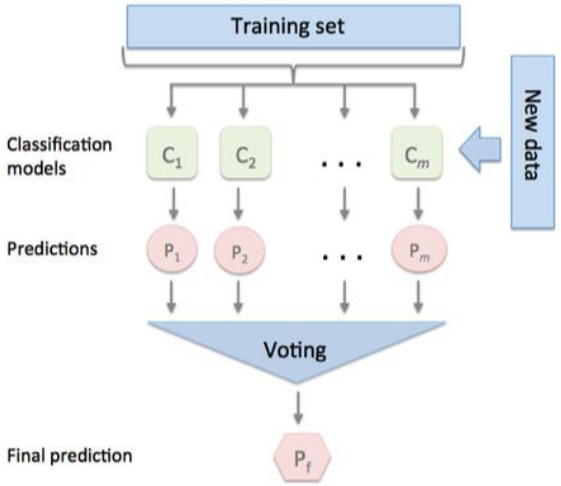
ENSEMBLE METHODS

WHAT ARE ENSEMBLE METHODS?

- Combining individual classifiers into a meta-classifier can realize better out-of-sample performance

- Generally, ensemble methods collect predictions from multiple models and use a voting mechanism to choose the best result





RANDOM FOREST

- Random Forest is a type of ensemble method built on decision trees
  - RF is simply a collection of decision trees
- A very simple adaptation but often results in drastic improvements to predictive accuracy
  - Good performance, scalable and easy to use
- Combining weak learners to build a strong learner
  - The idea that multiple independent models give better results than a single model
    - Assuming 100 classifiers that are each correct only 55% of the time
    - However, the probability that the majority of them are correct increases to 82% (cumulative binomial probability)
- Other ensemble methods: bagging and boosting

IMPLEMENTING RANDOM FOREST

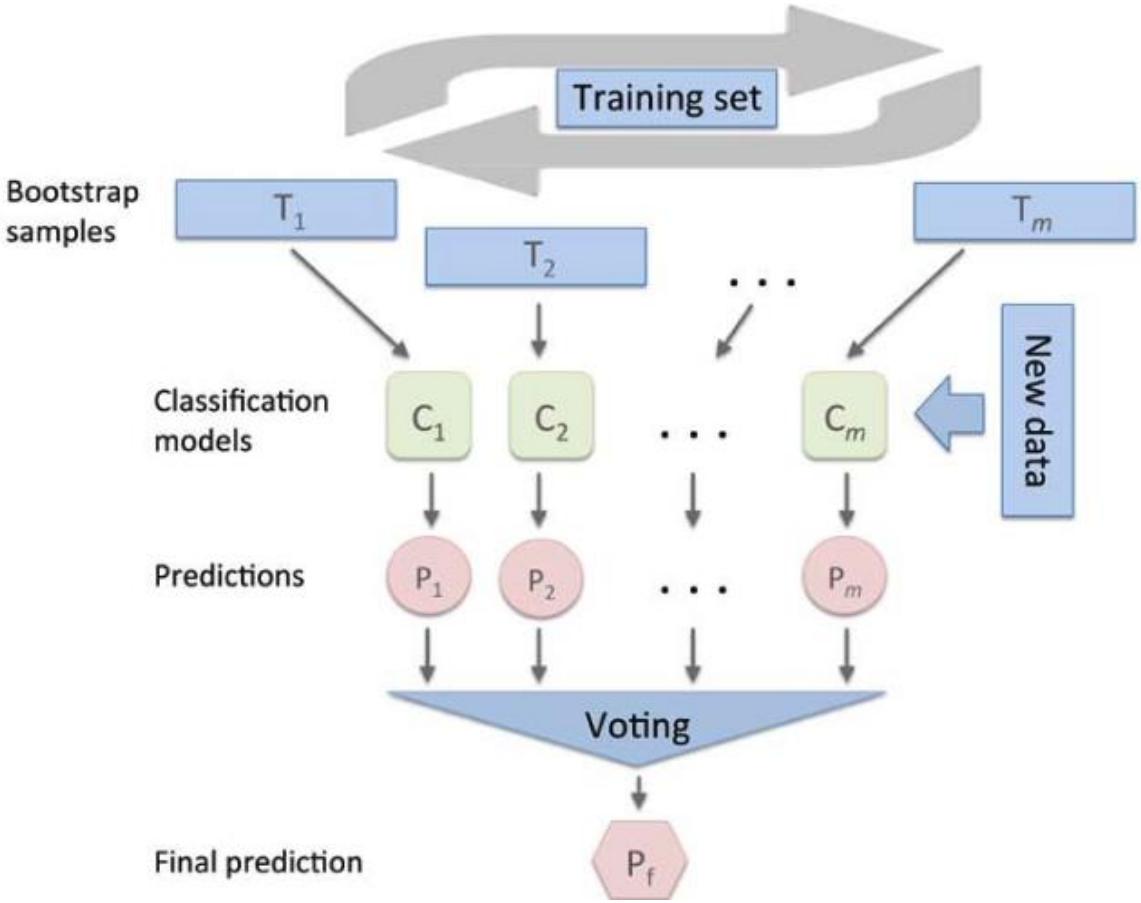
- Bootstrap a sample of size n
  - Bootstrapping is a method of sample selection where we make random selections with replacement
- Grow a decision tree using this sample
  - At each node, randomly select d features without replacement
  - Split this node using the best feature
- Repeat the above steps k times
- Aggregate the prediction by each tree and assign the class to the data via majority voting

SOME REMARKS

- Random forest model is a black box
  - Cannot be interpreted like a decision tree where you know that branches come out of important features
  - Random forest is a bunch of trees, and the prediction is a result of majority voting (i.e. most of the trees predict this outcome)

- However, RF has the following advantages
  - No need to worry about hyperparameter values
  - Don't need to prune the forest
  - Only need to choose the number of trees
    - The larger the number, the better the performance
    - However, the larger the number, the more expensive the computational cost

BAGGING – AN ALTERNATIVE TO MAJORITY VOTING

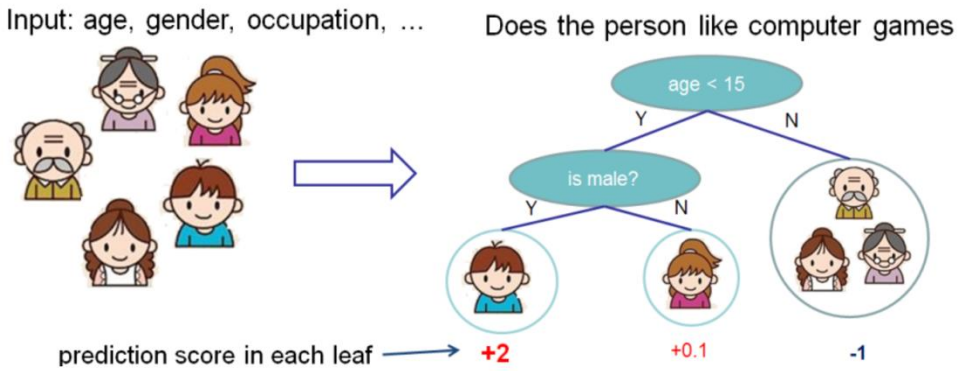


BOOSTING

- 3 commonly-used techniques
  - Adaptive boosting (adaboost)
    - A type of classifier boosting
  - Stochastic Gradient boosting
    - A type of regression boosting
  - Extreme gradient boosting (xgboost)
    - A type of tree ensemble (combination of classification and regression trees)
    - Fits a regularization variable in the objective function to prevent over-fitting
- The basic concept behind boosting is to start with very simple base classifiers (slightly better than random guessing)
  - These simple classifiers are called weak learners (e.g. 2-layer decision tree – they call this a stump)
  - Continuously make the weak learners learn from misclassified samples
    - Draw a random subset without replacement and train a weak learner
    - Draw a second subset without replacement, add 50% of the misclassifed samples in (1), and train another weak learner
    - Find the training samples that both (1) and (2) disagree, and train a third weak learner
    - Combine all 3 learners via majority voting
  - Note that boosting algorithms have a tendency to overfit

XGBOOST

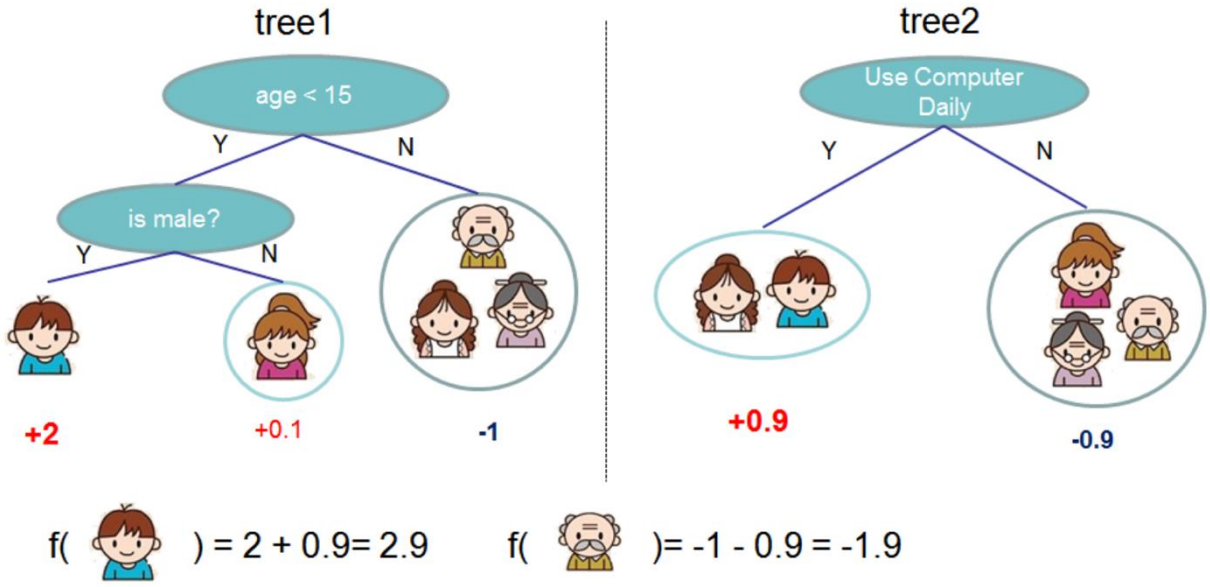
- Tree ensemble
  - Combination of classification and regression trees (CART)



  - In normal decision trees, the leaf contains the decision values
    - In CART, the leaves contain a real-valued score

TREE ENSEMBLE

- Example of 2 trees
  - Prediction scores are summed to get the final score
  - Note that the two trees complement each other



SOME COMMENTS ABOUT ENSEMBLE TECHNIQUES

- Non-trivial increases in computational complexity for (usually) modest improvements in predictive performance
- $1 million Netflix Grand Prize
  - For 3 years straight, thousands of teams could only improve upon the previous years results
    - Model tuning, factorization, simple ensembles etc
    - Could not beat the challenge of 10% improvement over the native Netflix recommendation algorithm
  - In 2009, a combined team (BellKors Pragmatic Chaos) beat the challenge using a complex blend of ensemble techniques
  - However, Netflix never implemented the winning model due to its complexity (development, execution and maintenance efforts that are not feasible for real-world application)

"[…] additional accuracy gains that we measured did not seem to justify the engineering effort needed to bring them into a production environment."

EXERCISE

- Upload "machine learning I.ipynb" into your Jupyter dashboard
  - Follow the comments and codes and feel free to try out new codes on your own
  - The visualization codes are meant to create graphs to help you picture the shape of the solution, don't crack your head trying to understand the codes!
    - The point is to help you build your intuition to what the ML model does

UNSUPERVISED LEARNING

UNSUPERVISED LEARNING

- Up till this point, we have been dealing with problems where we have an answer to (outcome)
  - All our regression equations are setup with an outcome variable on the LHS

- What happens when we suspect that there are relationships, subsets or clusters in our data, but have no answers upfront?

CLUSTERING

- Clustering is a category of unsupervised learning that helps us discover hidden structures and natural grouping in data based on a similarity function
  - K-means
  - Hierarchical
  - Density-based
  - Graph-based (basis of network theory)

- Other unsupervised learning algorithms
  - Association rules (Apriori)
  - Mixture models, Factor models, and Latent models

K-MEANS CLUSTERING

- The most widely used clustering algorithm in academia and industry
  - Easy to implement
  - Computationally efficient

- Concept
  - Find groups of similar objects based on a distance function that makes them more related to each other than to objects in other groups

- Issues
  - Requires us to specify number of clusters k
  - Inappropriate choice of k can result in poor clustering
    - We can determine the optimal number of k using elbow and silhouette plots
  - Clusters do not overlap and are not hierarchical
  - Clusters can be empty!

STEPS TO IMPLEMENT K-MEANS

- Randomly pick k centroids from the sample points
  - Remember that we have to manually specify the number of clusters k
- Assign every sample to its nearest centroid
- Move the centroids to the center of the samples
- Repeat steps 2 and 3 until
  - Cluster assignments no longer change
  - User defined threshold has been reached
    - Number of iterations
    - Error rate

DISTANCE MEASURE

- The key mathematical function in clustering is the distance function
  - This is a measure of similarity between objects
  - The nearer 2 objects are together, than to others, the more likely they belong together in a cluster

- The most commonly used distance function is the squared Euclidean distance
  - Note that in 2-d form, it's the same as how we measure straight-line distance between 2 objects
  - The generalized vector form is: $$d(\boldsymbol{x},\boldsymbol{y})^2 = \sum_{j=1}^{m}\left(x_j - y_j\right)^2 = \|\boldsymbol{x}-\boldsymbol{y}\|_2^2$$

- The objective is to minimize the within-cluster sum of squared errors

EXERCISE

- Upload "Machine Learning II.ipynb" into your dashboard
  - Follow the comments and instructions
  - Use the visualization to build your intuition as to the shape of the solution

MODEL EVALUATION AND HYPERPARAMETER TUNING

NEXT STEPS

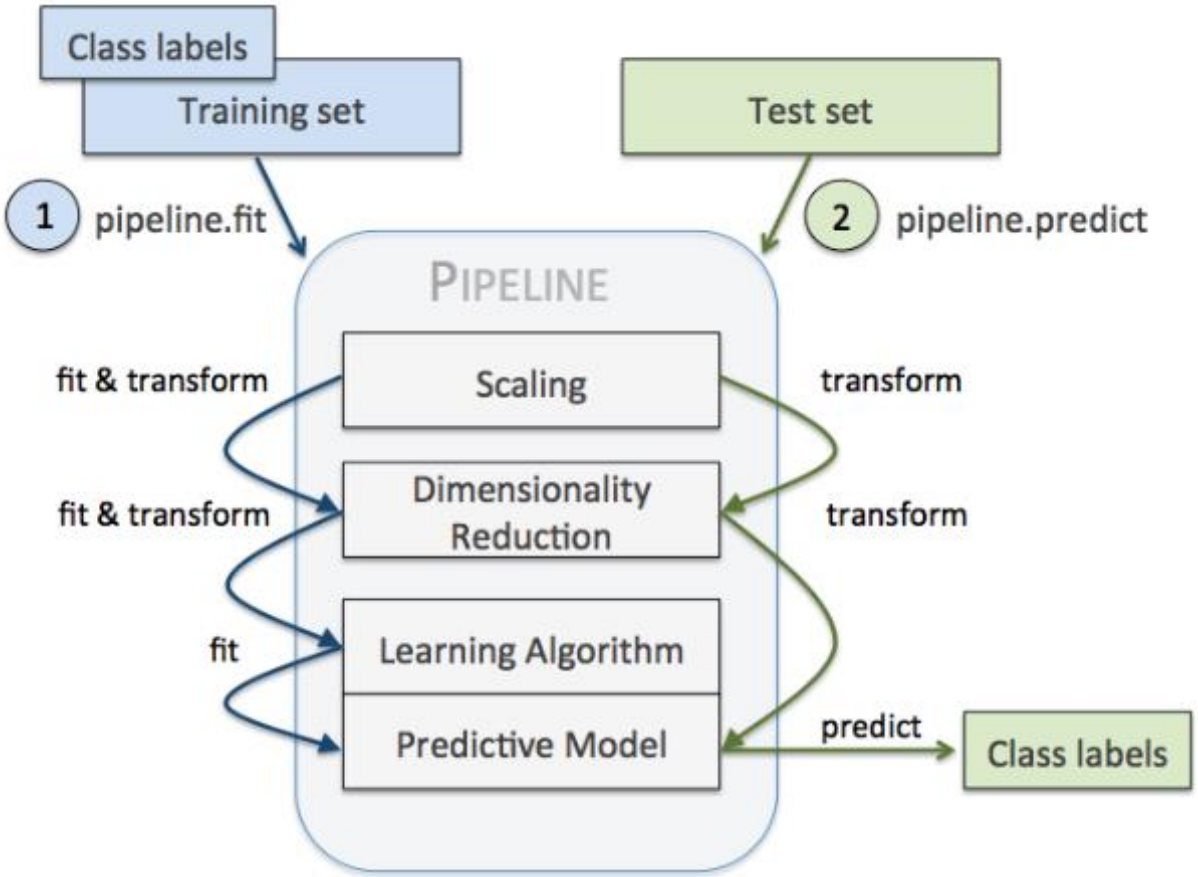
- Previous sections have familiarized us with the basics of machine learning algorithms
- Building good models, however, require more than just knowing when to use which algorithms
- We need to have a framework for evaluating model performances in response to algorithm fine-tuning
  - Some of these are best practices contributed by the data science community

USING PIPELINES TO STREAMLINE WORKFLOWS

- You may have noticed that we have been using and reusing multiple data preprocessing, compression and hyperparameters
  - Some of these processes require that we fit training and testing datasets with the same parameters (e.g. scaling and compression transformation)

- Scikit-learn comes with a very useful tool that enhances this workflow
  - Known as the Pipeline

PIPELINE WORKFLOW

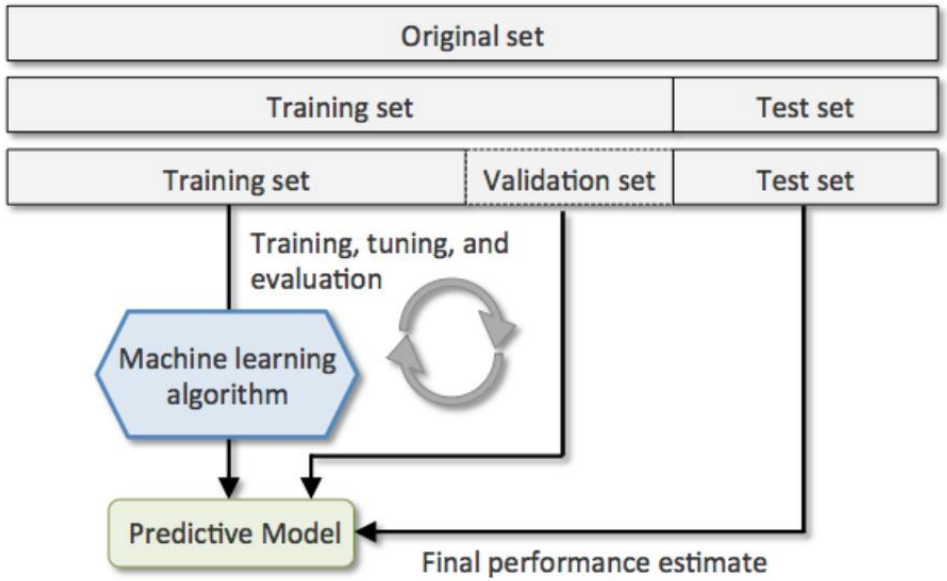


ASSESSING MODEL PERFORMANCE

- As covered in the previous sessions, model performance should be evaluated on 'never-seen-before' data or out-of-sample data
- Model performance can fall under the following 3 types:
  - Underfitting (model too simple, poor in-sample and out-sample performance)
  - Overfitting (model too complex, not generalizable and thus poor out-sample performance)
  - Good balance (this is what we want)
- We can use cross-validation techniques to figure out what the good balance is
  - Holdout method
  - K-fold

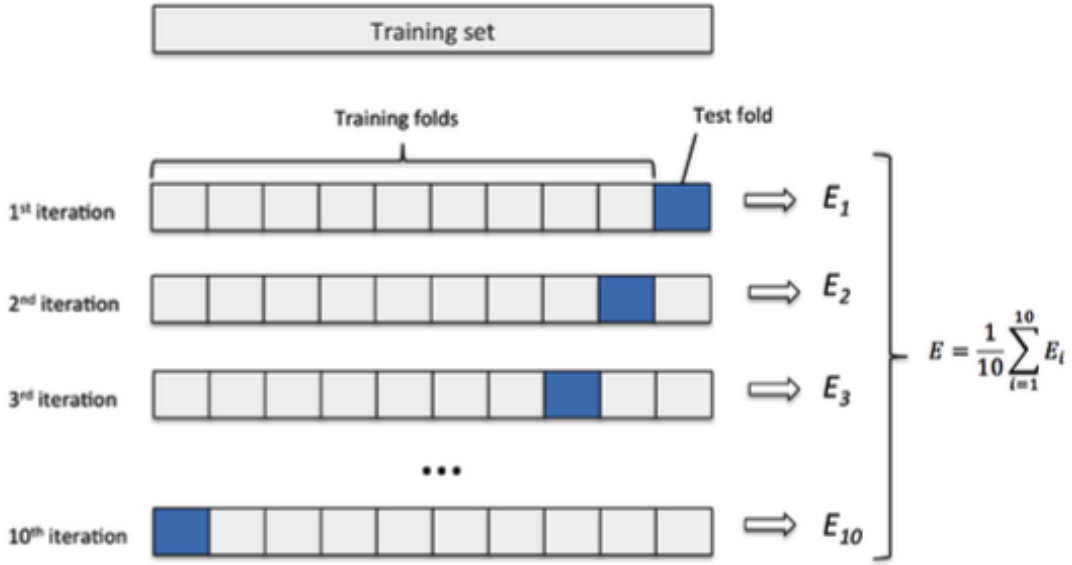
THE HOLDOUT METHOD

- Split the data into 3 sets:
  - Training set: used to fit the models
  - Validation set: models with the best performance on this set will be selected
  - Test set: final performance estimate
- However, performance estimates are sensitive to how we partition the data
  - Not the best CV method, though many still use this



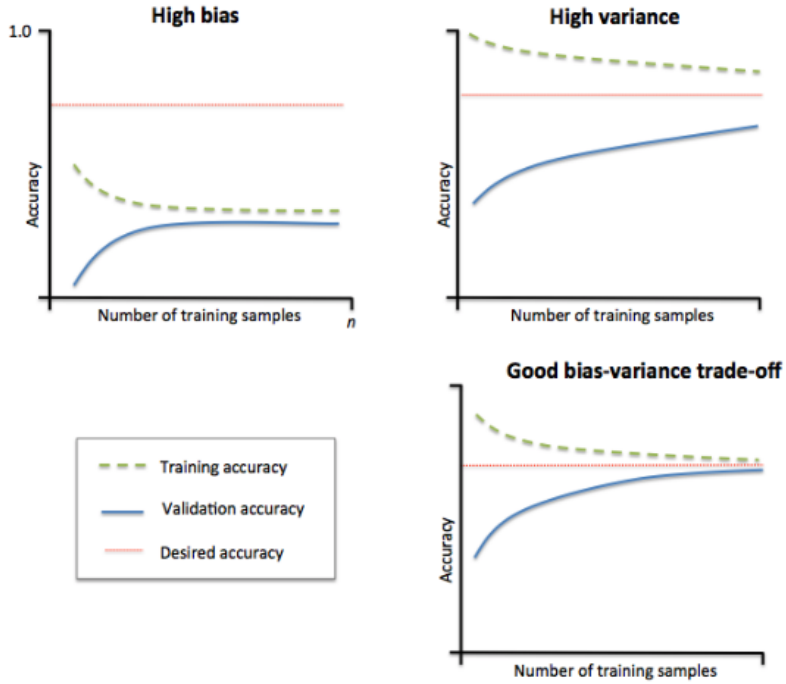
K-FOLD CROSS VALIDATION

- The preferred method for performance evaluation
  - 10 folds is the standard
  - But we will need to increase the number of folds for smaller and smaller datasets (up to leave-one-out for very small datasets)
    - Note that this will increase computational time
    - Also, the training samples will be pretty much similar to each other (the fitted models don't differ much from each other)
- To handle unequal class proportions (which can be a huge issue if the inequality is large), we can preserve the proportions in each fold
  - This is known as the stratified k-fold cross-validation



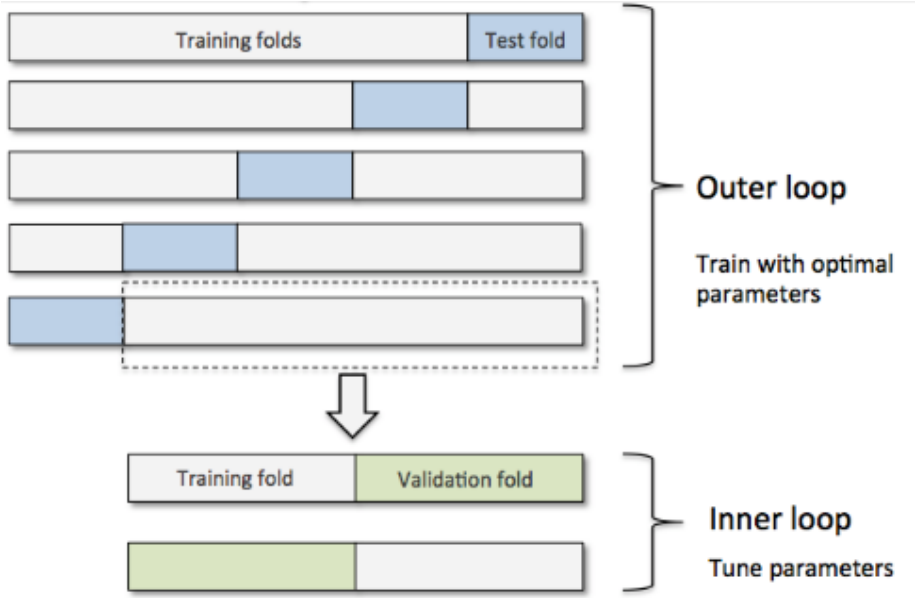
PERFORMANCE DIAGNOSTICS

- We can use learning and validation curves to diagnose our results
- Learning curves
  - Estimate underfitting vs overfitting
  - Graph on upper-left has low training and cv accuracy (underfitting)
    - Increase variables through collection or transformation
    - Decrease regularization (increasing c)
  - Graph on upper-right has high training but low cv accuracy (overfitting)
    - Increase observations
    - Decrease complexity through feature selection or extraction
    - Increase regularization (decreasing c)
- Validation curves
  - Improve models by varying parameters



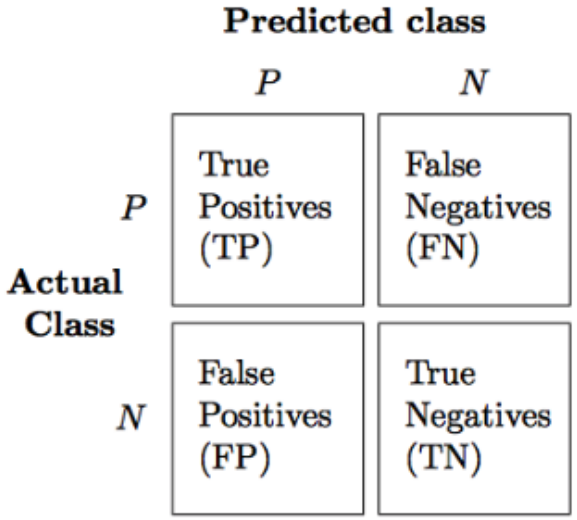
OTHER USEFUL TECHNIQUES

- Grid search
  - Similar to validation curves, we specify a list of values for each hyperparameter
  - This method finds the optimal combination of hyperparameter values in a brute force exhaustive manner
- Nested cross-validation
  - Performing this test on different ML algorithms can aid us in choosing the best candidate model
  - Exhibits good performance in academic papers



CONFUSION MATRIX

- We can drill deeper into the accuracy measure to evaluate a model's precision and recall
- Precision, recall and the associated F1-score is represented by a confusion matrix
  - The % of true and false positive/negative
  - Precision is the % of true positives over true and false positives
    - How many of the predicted positives are indeed true?
  - Recall is the % of true positives over true positives + false negatives
    - How many of the actual positives are correctly predicted?
  - The F1 score is 2 x (Precision x Recall)/(Precision + Recall)



EXERCISE

- Upload "Tuning and Performance.ipynb" and "wdbc.csv" into your dashboard
  - Follow the comments and instructions

- As part of a final exercise, we will let you explore a financial dataset using the techniques that we have learnt
  - Upload "key_financials.xlsx" into your Jupyter dashboard
  - Upload "financial_exercise.ipynb" into your Jupyter dashboard
  - Follow the comments and codes and attempt the questions

QUESTIONS?

Email any queries to
jackhong@smu.edu.sg